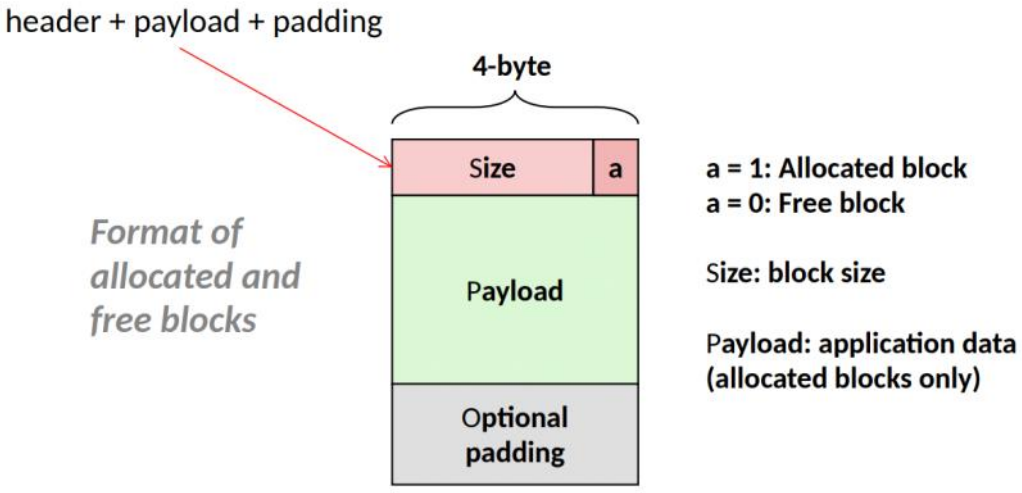
Dynamic Memory Allocation



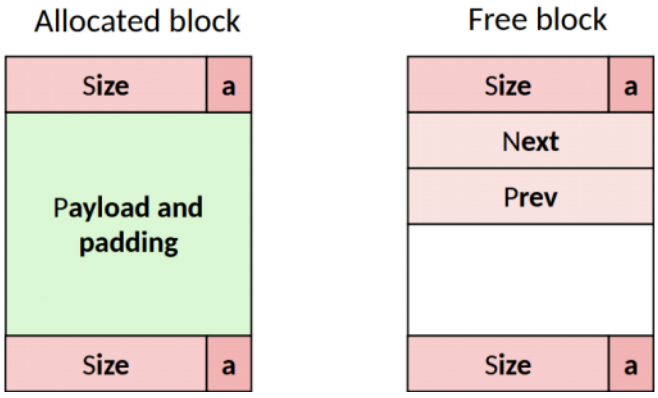
In 32-bit: 8-byte alignment
Header -- 4 byte, store size of the whole block, last bit show allocated or free
Payload -- data
Padding -- exist for 8-byte alignment

In 64-bit: 16-byte alignment
Header -- 8 byte, store size of the whole block, last bit show allocated or free
Payload -- data
Padding -- exist for 16-byte alignment

- *First fit:*
  - Search from beginning, choose ***first*** free block that fits:
- *Next fit:*
  - Like first fit, except search starts where previous search finished
- *Best fit:*
  - Search the list, choose the ***best*** free block: fits, with fewest bytes left over (i.e. pick the smallest block that is big enough for the payload)
  - Keeps fragments small
  - Will typically run slower than first fit

Slower, but improve the efficiency of using memory

Explicit Free Lists



Freeing With Explicit Free Lists

- Where in the free list to put a newly freed block?
  - Insert freed block at the beginning of the free list (LIFO)
    - *Pro:* simple and constant time
  - Insert freed blocks to maintain address order:

    $addr(prev) < addr(curr) < addr(next)$
    - *Pro:* may lead to less fragmentation than LIFO